Agenda - Exercises

Testing expressions, bootstrapping

Expressions to VM Code

VM expressions to Assembly

Representing Arrays

Representing Objects

Demo: Testing expressions

```
//VM Code
push constant 7
push constant 8
add
```

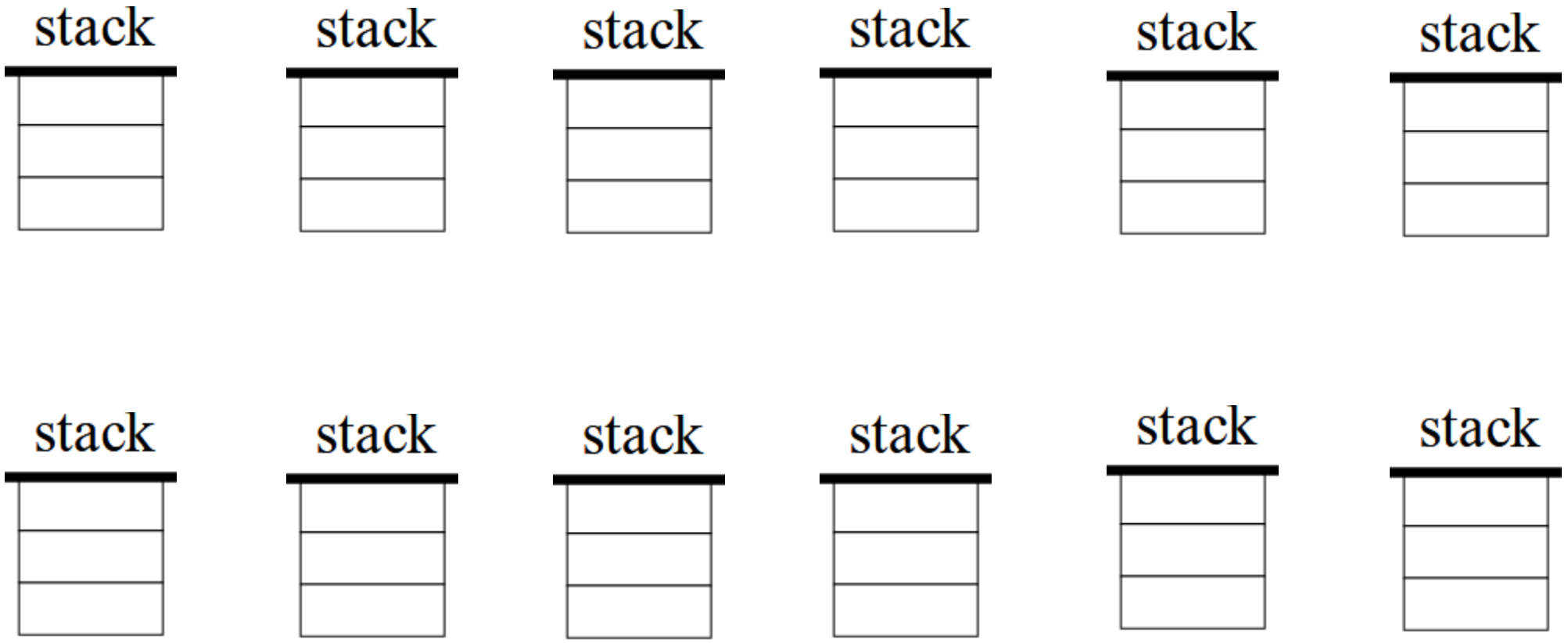
```
// Assembly
// Pseudocode
```

```
// Assembly
// Pseudocode
// with bootstrap
```

Convert the expression to VM Code

Express with vm code: (4-5) – (3+7)

Visualize the stack execution



Convert the loop to VM Code

```
void loop(int n) {
   int i
   while (i < n) {
      i++
   }
}
```

Compute assembly pseudocode

```
label loop
  push local 0
  push argument 0
  eq
  if-goto end
  push local 0
  push constant 1
  add
  goto loop
label end
  ...
```

Locals

0

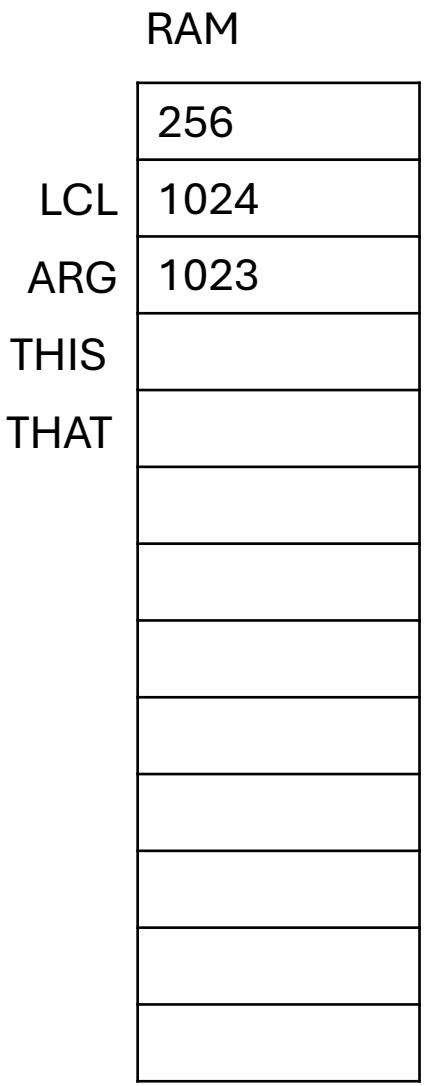
Arguments

3

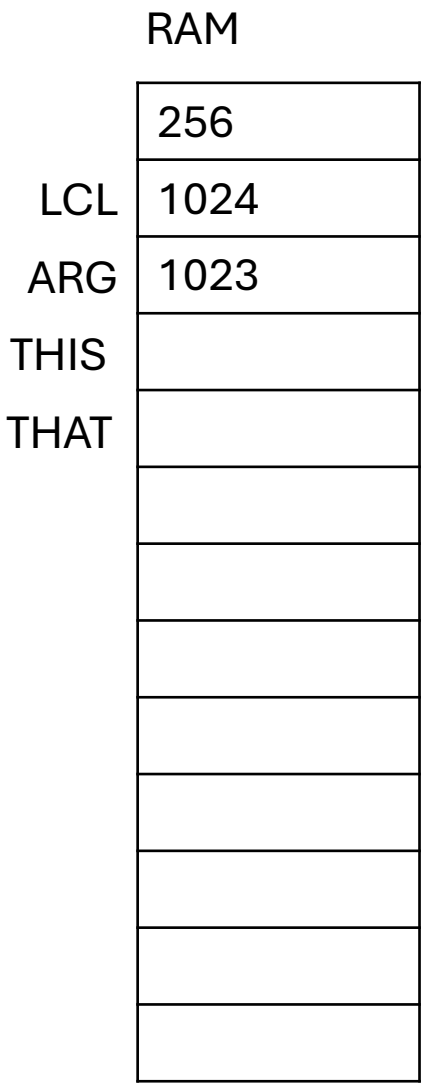
RAM

	256
LCL	1024
ARG	1023
THIS	
THAT	

Compute assembly



Visualize assembly



VM Arrays

```
// High-level Code
int* bar = {0,2,4,6,8};
bar[1] = 3;
```

What is the equivalent pointer expression for bar[1] = 3?

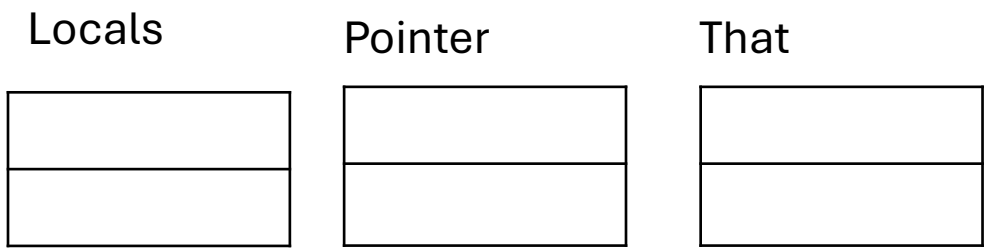
Suppose bar is located at address 890 in RAM and has the value 1024. Draw the array in RAM.

Convert the array assignment to VM Code

```
// High-level Code
int* bar = {0,2,4,6,8};
bar[1] = 3;
```

Write VM code and visualize its virtual memory segments

```
// High-level Code
int* bar = {0,2,4,6,8};
bar[1] = 3;
```



Convert the object function call to Assembly

```
// High-level code

class Vector {
  int x;
  int y;
....
  void reverse() {
    x = -x;
    y = -y;
   }
}

Vector dir;
```

Suppose dir is located
at address 600 in RAM
and has the value 890.
Draw the object in RAM.

Write VM code for reverse

```
// High-level code

class Vector {
  int x;
  int y;
....
  void reverse() {
    x = -x;
    y = -y;
   }
}

Vector dir;
dir.revers();
```

Visualize your VM code

```
// High-level code

class Vector {
  int x;
  int y;
....
  void reverse() {
    x = -x;
    y = -y;
  }
}

Vector dir;
dir.revers();
```